Definition: Two edges e and f are in the same biconnected component if there is a simple cycle that contains both e and f.

The following algorithm takes an undirected graph as input and labels each edge so that all edges in the same biconnected component have the same label. It uses back edges to identify cycles. When a back edge from v to w is identified, all edges on the path from w to v in the tree are known to be in the same biconnected component. In addition, any back edges connecting two vertices on that path are also included. The algorithm keeps track of the vertex discovered earliest on any path in the tree. This is accomplished by the variable $low[v]$.

```
▷ Assume that the graph is connected; otherwise find connected components first.
▷ The global variable time is initialized to 0.
▷ All edges are labeled as not discovered.
▷ edgeStack contains edges in the corrent working biconnected component.
▷ Then the following (recursive) procedure is called with an arbitrary start vertex v
▷  and a null parent p.
function BICON(v, p) is
    ▷ returns the earliest discovery time for back edges in the subtree rooted at v
    mark v as discovered
    increment time; discoverTime[v] ← time
    ▷ back keeps track of the earliest discovery time for back edges in the subtree rooted at v
    back ← discoverTime[v]
    for all edges vw incident to v do
        if w is not discovered then      ▷ tree edge
            push vw onto edgeStack
            low ← BICON(w, v)
            if low ≥ discoverTime[v] then      ▷ end of component
                pop everything on edgeStack up to and including vw = wv
                  and make them the edges of a new component
            endif
            back ← min(low, back)
        else if discoverTime[w] < discoverTime[v]  and w ≠ p then
            ▷ back edge, but not to parent
            push vw onto edgeStack
            back ← min(low, back)
        endif
    end do
    return back
end BICON
```